Amendments to the Claims

A complete list of pending claims follows, with indicated amendments:

1. (Previously Amended) A method for establishing a device driver in an open source operating system, comprising the steps of:

providing a device driver having at least one module in executable form and a service layer in open source form; and

compiling the service layer against the kernel of the open source operating system after each modification to the kernel of the open source operating system;

wherein the compiled service layer acts as an interface between the kernel of the operating system and the at least one executable module of the device driver.

2. (Original) The method for establishing a device driver in an open source operating system of claim 1, wherein the step of compiling the service layer against the kernel comprises the step of associating the naming convention of function calls in the kernel to the naming convention of expected function calls in the device driver.

3. (Original) The method for establishing a device driver in an open source operating system of claim 1, further comprising the step of linking the compiled service layer to the at least one module in executable form to form the device driver.

4. (Original) The method for establishing a device driver in an open source operating system of claim 3, further comprising the step of storing the device driver in memory.

5. (Original) The method of claim 1, wherein the step of providing a device driver comprises the step of providing a device driver having multiple modules in executable form, each of the modules associated with a hardware architecture of a computer system.

6. (Original) The method for establishing a device driver in an open source operating system of claim 5, further comprising the step of linking the compiled service layer to the multiple modules in executable form to form the device driver.

7. (Original) The method for establishing a device driver in an open source operating system of claim 6, further comprising the step of storing the device driver in memory.

8. (Previously Amended) A computer system comprising:

a processor;

a memory;

an open source operating system having a kernel;

a device driver, the device driver comprising,

an executable module compiled from an open source service layer, and

at least one executable module,

wherein the executable module compiled from the open source service layer provides an interface between the kernel of the operating system and the at least one executable module such that the executable module compiled from the open source service layer receives kernel-specific function calls from the kernel of the operating system, and wherein the

executable module is compiled from the open source service layer following each modification to the kernel of the operating system.

9. (Original) The computer system of claim 8, wherein the device driver is loaded in memory of the computer system.

10. (Original) The computer system of claim 8, wherein the device driver comprises multiple executable modules and wherein each of the executable modules is associated with a hardware architecture of a computer system.

11. (Original) The computer system of claim 8, wherein the kernel of the operating system and the executable module compiled from the open source service layer send and receive function calls according to the same naming convention.

12. (Original) The computer system of claim 11, wherein the name convention comprises the use of a suffix for the naming of function calls, the suffix providing a naming convention that is specific to the kernel of the operating system.

13. (Previously Amended) A method for loading a device driver in a computer system having an open source operating system, comprising the steps of:

compiling an open source service layer against the kernel of the operating system following a modification to the kernel of the operating system; and

linking the compiled service layer to a set of precompiled driver modules, each of the precompiled driver modules being associated with a hardware architecture of a computer system;

wherein the compiled service layer provides an interface between the kernel of the operating system and the precompiled driver modules.

14. (Original) The method for loading a device driver of claim 13, wherein the kernel of the operating system and the compiled service layer is operable to send and receive function calls that are named according to the same naming convention.

15. (Original) The method for loading a device driver of claim 13, further comprising the step of recompiling the open source service layer if it is determined that the kernel of the open source service layer has been modified.

16. (Previously Amended) The method for loading a device driver of claim 13, further comprising the step of linking the recompiled service layer to the set of precompiled driver modules.

17. (Original) The method for loading a device driver of claim 13, further comprising the step of determining, prior to compilation of the open source service layer, whether a precompiled device driver exists that is associated with the kernel of the operating system and loading the precompiled device driver if such a device driver exists.

18. (Original) The method for loading a device driver of claim 13,

wherein the function calls passed between the kernel of the operating system and the compiled open source service layer are not specific to the hardware architecture of the computer system; and

wherein the function calls passed between the compiled open source service layer and the precompiled driver modules are specific to the hardware architecture of the computer system.

19. (Previously Amended) The method for loading a device driver of claim 13, further comprising the steps of,

recompiling the service layer if it is determined that the kernel of the operating system has been modified; and

relinking the recompiled service layer to the set of precompiled driver modules.

20. (Original) The method for loading a device driver of claim 19, wherein the recompiled service layer is operable to send and receive function calls that are named according to the same naming convention.